### **Assignment Title**

### ***"Design and Implement a Gym Management System (GMS)"***

### **Objective**

Develop a Python-based **Gym Management System (GMS)** to efficiently manage:

1. Member registration and subscription tracking.
2. Role-based access for Admin, Trainer, and Member.
3. Workout schedules, subscription plans, and attendance tracking.

### **Features and Deliverables**

#### **1. Roles and Access Control**

* **Admin**:
  + Manage gym facilities, trainers, and subscription plans.
  + Add, update, and delete member profiles.
  + Generate revenue and attendance reports.
* **Trainer**:
  + Assign workout schedules to members.
  + Track member attendance and progress.
* **Member**:
  + View workout schedules and update progress.
  + Renew subscriptions and track their status.

#### **2. Core Functionalities**

1. **Membership Management**:
   * Add, update, and delete members.
   * Track active and expired memberships.
2. **Workout Schedule Management**:
   * Create and assign workout schedules to members.
   * Allow members to view and update progress.
3. **Attendance Tracking**:
   * Track daily attendance for members.
   * Provide attendance reports for trainers and admins.
   * Generate attendance summaries for trainers and admins.
4. **Calculate**:
   * Total revenue from subscriptions.
   * Number of active and expired memberships.
   * Attendance percentages for members.
   * Top-performing members based on workout progress.
5. **Data Persistence**:
   * Save and reload all data (members, schedules, subscriptions, attendance) using CSV or JSON.
   * Export reports to CSV for sharing.

#### **3. Submission Requirements**

1. **Code**: A complete Python implementation of the GMS.
2. **Documentation**:
   * Provide a flowchart to show user interactions with the system.
   * Include a description of classes and methods in the program.
3. **Screenshots**:
   * Demonstrating role-based interactions, schedule creation, and attendance tracking.

### **Evaluation Criteria**

1. **Functionality and Coverage** (40%):
   * Includes membership management, role-based access, and attendance tracking.
2. **Code Quality** (20%):
   * Readable, modular, and adheres to OOP principles.
3. **Documentation** (20%):
   * Well-explained clear flowchart and description of classes/methods in the program.
4. **Creativity and Features** (20%):
   * Additional features like progress tracking or reward systems.